CLAIMS

What is Claimed is:

1. A method comprising:

stalling a heap allocation function call to a heap allocation function originating from a request by an application for a block of heap buffer;

predicting a predicted block of said heap buffer to fulfill said request; and

determining if a forward link (F-link) and a backward link (B-link) of said predicted block are addresses within a heap segment associated with said predicted block.

2. The method of Claim 1 further comprising hooking said heap allocation function.

3. The method of Claim 1 further comprising determining a size of said block.

4. The method of Claim 3 wherein said predicted block has said size.

5. The method of Claim 3 wherein a freelist comprises a plurality of free blocks having said size, said predicted block being on said freelist.

6. The method of Claim 1 wherein said predicted block is on a predicted freelist.

7. The method of Claim 6 further comprising determining whether a F-link of a predicted list head of said predicted freelist points into said heap segment.

8. The method of Claim 6 further comprising determining whether a B-link of a predicted next block of said predicted freelist points into said heap segment.

9. The method of Claim 1 wherein upon a determination that said F-link and said B-link of said predicted block are not addresses within said heap segment, said method further comprising taking corrective action.

10. The method of Claim 9 wherein said taking corrective action comprises setting said F-link and said B-link to be an address of a list head of a freelist comprising said predicted block.

11. A method comprising:

stalling a heap deallocation function call to a heap deallocation function originating from a release by an application of a block of heap buffer, wherein said block is a deallocation block that is being deallocated to a deallocation freelist; and

determining if a forward link (F-link) of a list head of said deallocation freelist and a backward link (B-link) of a first block of said deallocation freelist are addresses within a heap segment associated with said deallocation freelist.

12. The method of Claim 11 further comprising reading said F-link and said B-link.

13. The method of Claim 11 further comprising hooking said heap deallocation function.

14. The method of Claim 11 further comprising determining said block.

15. The method of Claim 11 wherein upon a determination that said F-link and said B-link are addresses within said heap segment, said method further comprising releasing said heap deallocation function call.

16. The method of Claim 11 wherein upon a determination that said F-link or said B-link are not addresses within said heap segment, said method further comprising taking corrective action.

17. The method of Claim 11 wherein said F-link or said B-link is a stray F-link or stray B-link, said method further comprising determining if said stray F-link or stray B-link is a known false positive.

18. The method of Claim 11 further comprising determining if said block is to be coalesced with other free blocks.

19. The method of Claim 11 wherein said block is to be coalesced with a coalesced block, said method further comprising:

determining if a F-link and a B-link of said coalesced block are addresses within a heap segment associated with said coalesced block.

20. The method of Claim 19 further comprising determining if there are other blocks to be coalesced with said block.

21. A method comprising:

determining that a freelist comprising a list head, a first free block and a second free block is corrupt; and

determining if said list head is corrupt, wherein upon a determination that said list head is corrupt, said method further comprising:

setting a forward link (F-link) and a backward link (B-link) of said list head to an address of said list head.

22. The method of Claim 21 wherein upon a determination that said list head is not corrupt, said method further comprising:

determining if said first free block is corrupt.

23. The method of Claim 22 wherein upon a determination that said first free block is corrupt, said method further comprising:

setting a F-link and a B-link of said first free block to said address of said list head; and

setting said B-link of said list head to an address of said first free block.

24. The method of Claim 22 wherein upon a determination that said first free block is not corrupt, said method further comprising:

setting a F-link of said second free block to said address of said list head;

setting a B-link of said second free block to an address of said first free block; and

setting said B-link of said list head to an address of said second free block.

25. A computer-program product comprising a computer-readable medium containing computer program code comprising:

a heap buffer overflow exploitation prevention application for stalling a heap allocation function call to a heap allocation function originating from a request by an application for a block of heap buffer;

said heap buffer overflow exploitation prevention application further for predicting a predicted block of said heap buffer to fulfill said request; and

said heap buffer overflow exploitation prevention application further for determining if a forward link (F-link) and a backward link (B-link) of said predicted block are addresses within a heap segment associated with said predicted block.